Performance evaluation targeting Quality of Experience

EPEW 2018

G. Rubino

INRIA Rennes, France

October 30, 2018

Abstract

- At Inria, we have been developing tools for measuring QoE in apps and services built on top of the Internet for years.
- This concerns apps and services that focus on transporting video or audio or voice content (which represent most of the Internet traffic).
- Our basic tools are G-networks used as Machine Learning (ML) tools.
- The result was our PSQA (Pseudo-Subjective Quality Assessment) technology. It works in real time, for one-way and for bi-directional comms, and it's accurate ("as accurate as possible").
- This approach can also be used to do performance evaluation analysis targeting QoE instead of classic metrics such as delays, losses, backlogs, etc., even following an analytical approach.

On PSQA

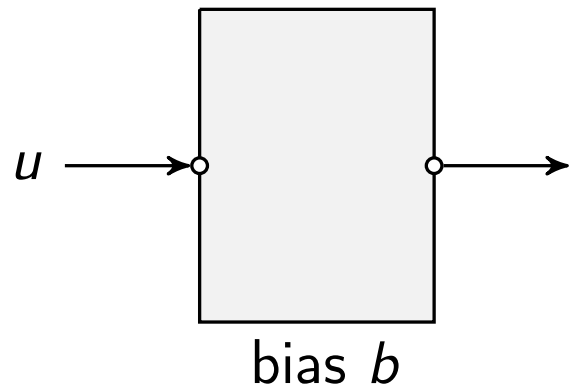
- The idea is to build a QoE metric based on specific QoS measures plus channel parameters (called the input variables), using Supervised Learning, the data coming from subjective tests, that is, from panels of human observers.
- In a nutshell, we build a database of, say, video sequences having variable qualities because of different combinations of the input variables, whose quality is then evaluated by the panels. Then, ML builds the mapping (I usually denote it $\nu()$) from the input variables into quantified QoE, ignoring the signals' content.
- Since the ML tool is a G-network, $\nu()$ is a rational function mapping the QoS and channel measures, the input variables, into QoE (Perceptual Quality in this case), the output variable.

Outline

Outline

Classic artificial neurons

- A "classic artificial neuron" can be seen as a parametric real function of a real variable:

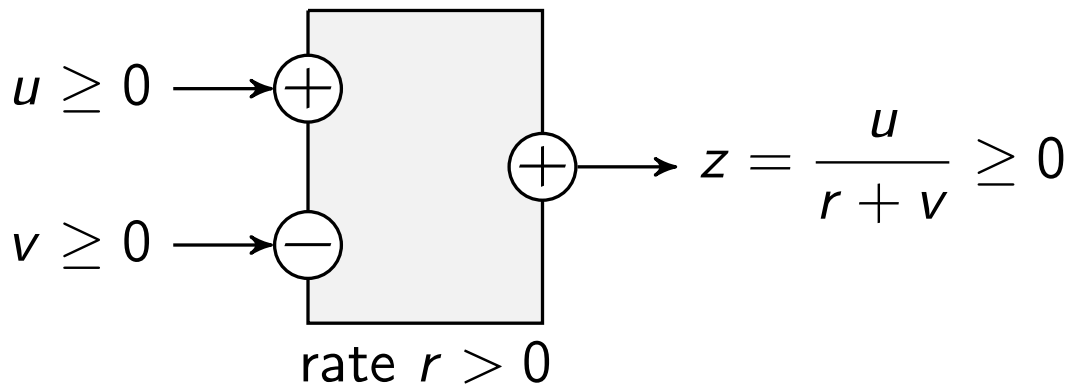


$$\text{for instance } z = \frac{1}{1 + \mathrm{e}^{-u+b}}$$

- We consider b as a parameter of the function implemented by the neuron (called *activation* function).
- There are many different activation functions used in practice: $z = \tanh(u - b)$, $z = 1\big(u \geq b\big)$, ReLU...

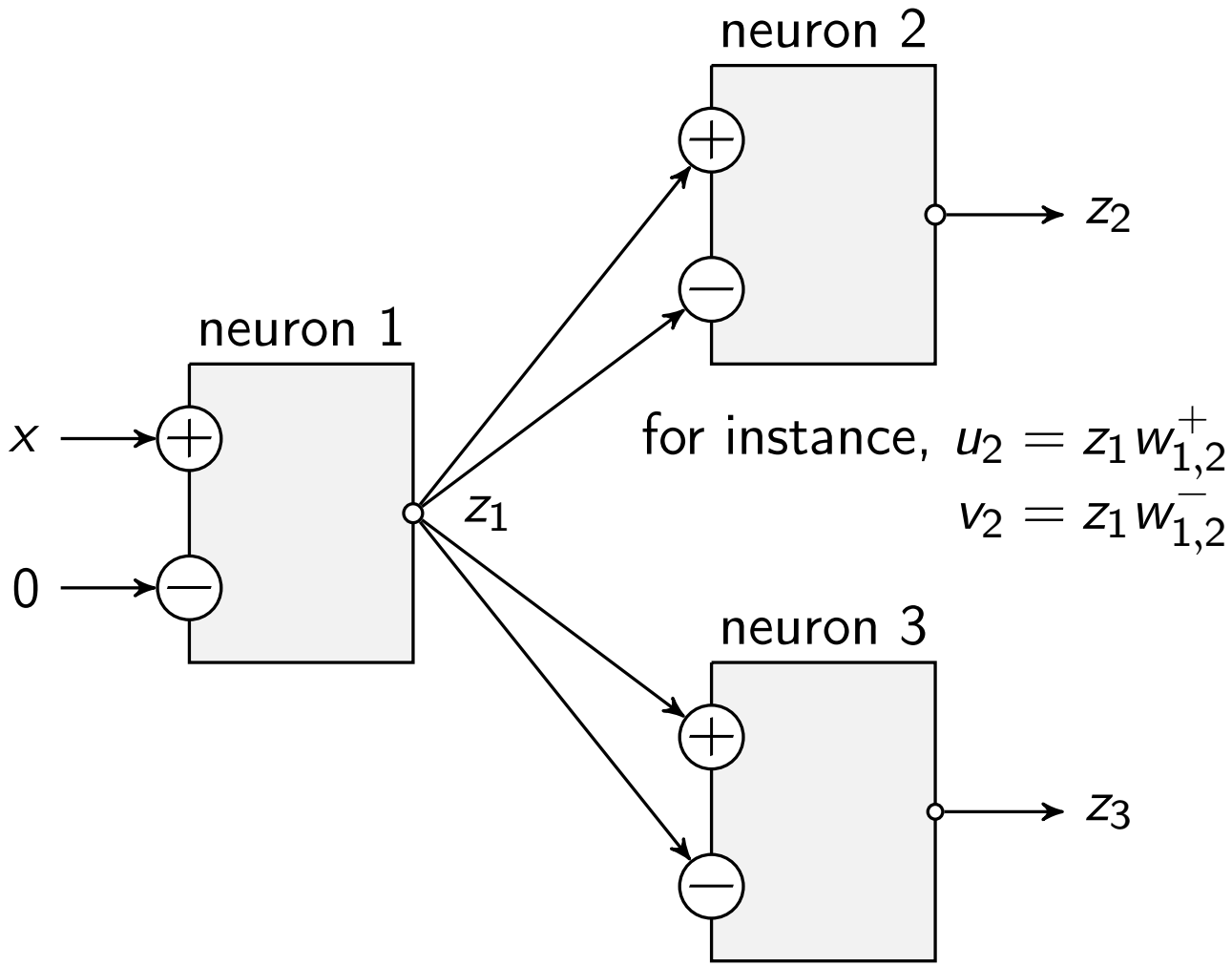
Random Neurons

- A Random Neuron is a parametric positive real function of two real positive variables (we also say "ports") called "positive" and "negative" ports.

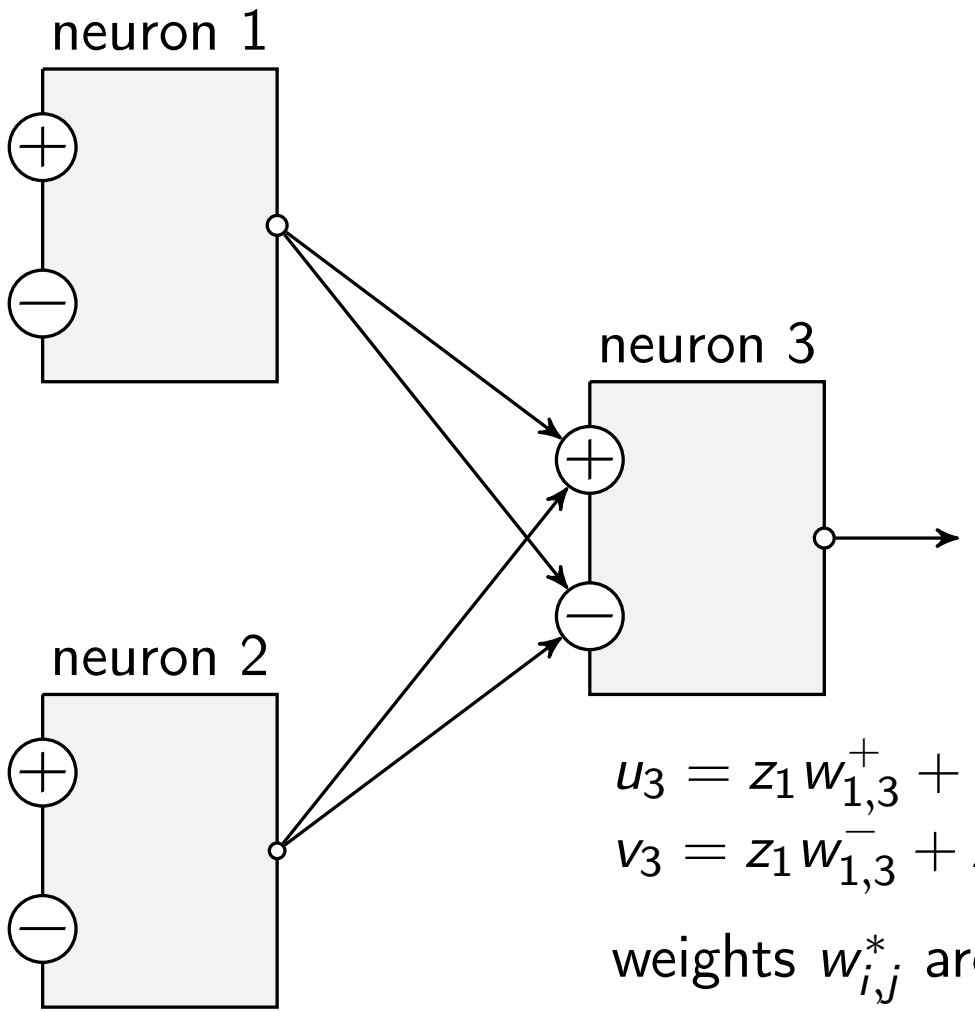


- We see the number $r > 0$ as a parameter (the *rate* of the neuron).
- There are several variants of this model. In the main and original one, we must use the function $z = \min(u/(r+v),\ 1)$.
- Inventor: E. Gelenbe, Imperial College, in the late 80s.
- Observe that there is nothing random here.

A Random Neural Network (RNN) is a set of interconnected RNs. The connections are weighted by nonnegative reals.



Jonctions at input ports are additive:



$$u_3 = z_1 w^+_{1,3} + z_2 w^+_{2,3}$$
$$v_3 = z_1 w^-_{1,3} + z_2 w^-_{2,3}$$

weights $w^*_{i,j}$ are nonnegative

Possible restriction

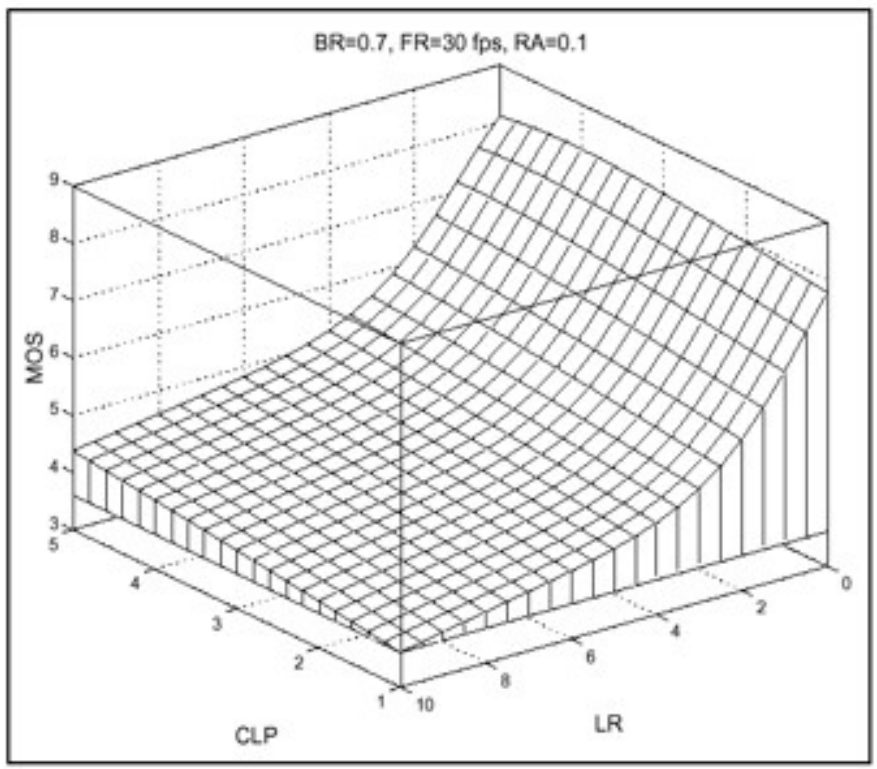
- Because of the queuing origin of the RNN tool, in order to respect the strict meaning of the model we must have

$$\text{for each neuron } i, \quad \sum_j \left(w_{i,j}^+ + w_{i,j}^-\right) = r_i.$$

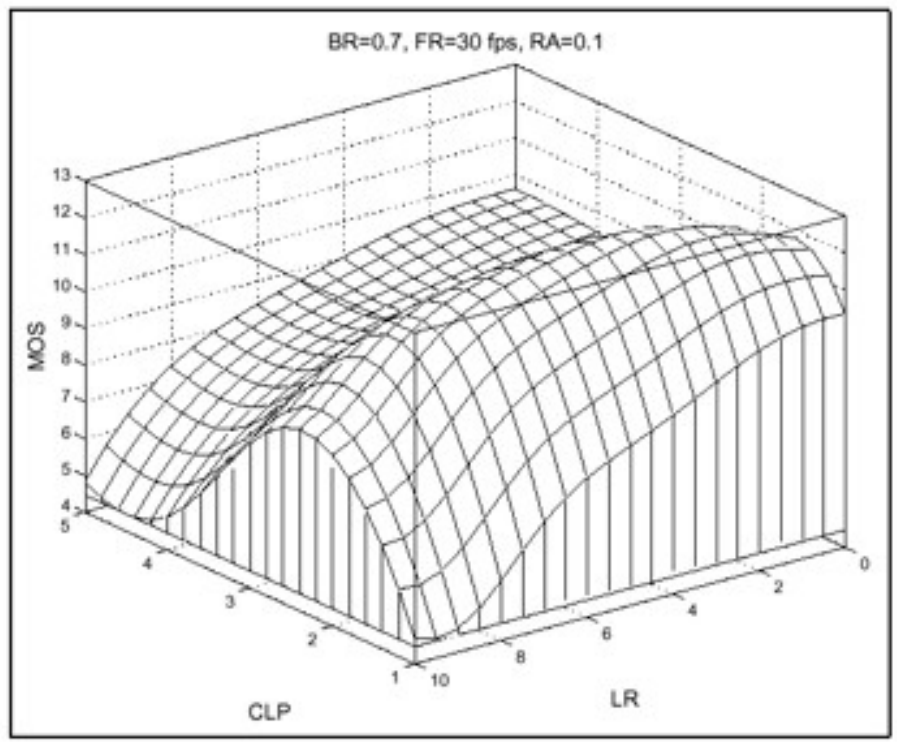
- This is a constraint to be respected in order to keep the queueing interpretation (and thus, to access all the theory and algorithms associated with).
- The same reason explains the use of $z = \min(u/(r+v),\ 1)$ instead of $z = u/(r+v)$ (more on this below).

Why using RNNs in our PSQA project?

Because of the favorable comparison with standard software (at the time, several years ago). An example: with the same data and with the same cost (more precisely, the same number of weights), RNN on the left, Matlab on the right (an old version of its ToolBox on ML).



(a) Correctly trainined



(b) Example of an over-trained ANN

Outline

Origin of the model

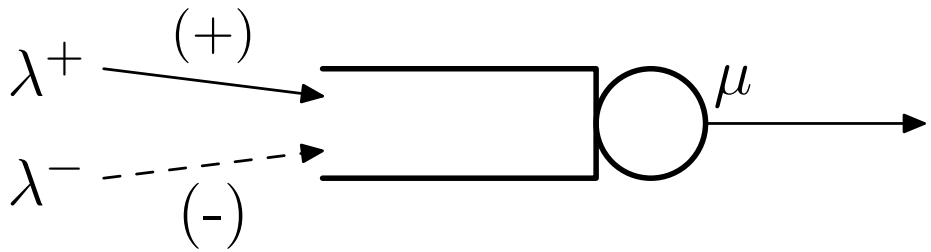
Refresher:

Consider the M/M/1 model, with arrival rate λ and service rate μ. The queue is stable iff $\lambda < \mu$; in that case, the steady state distribution $(\pi_n)_{n\geq 0}$ is given by $\pi_n = (1-\rho)\rho^n$, $n \in \mathbb{N}$, where the number $\rho = \lambda/\mu = 1 - \pi_0$ is the utilization factor, or load, of the system.

- $\rho = \Pr($ system busy at infinity $)$; if $\lambda \geq \mu$, then we have that $\Pr($ there exists $t < \infty$ such that after t, system is always busy $) = 1$.
- See then that the load of the queue at infinity, ρ, satisfies

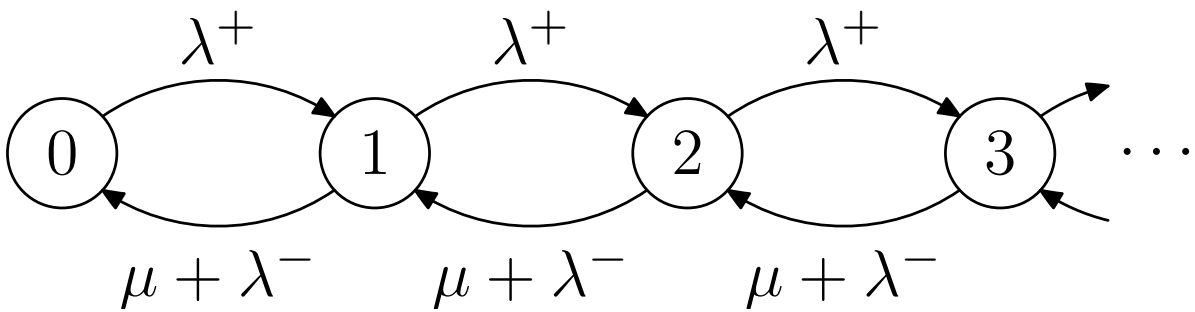
$$\rho = \min\{\frac{\lambda}{\mu},\ 1\}.$$

A G-queue



A basic G-queue: positive (standard) customers arrive with rate λ^+; negative ones arrive with rate λ^-; both arrival processes are Poisson; the service rate is μ. Add usual independence conditions. To understand the semantics, see the graph below.

We have stability $\iff \lambda^+ < \mu + \lambda^-$, and there, $\rho = \dfrac{\lambda^+}{\mu + \lambda^-}$.



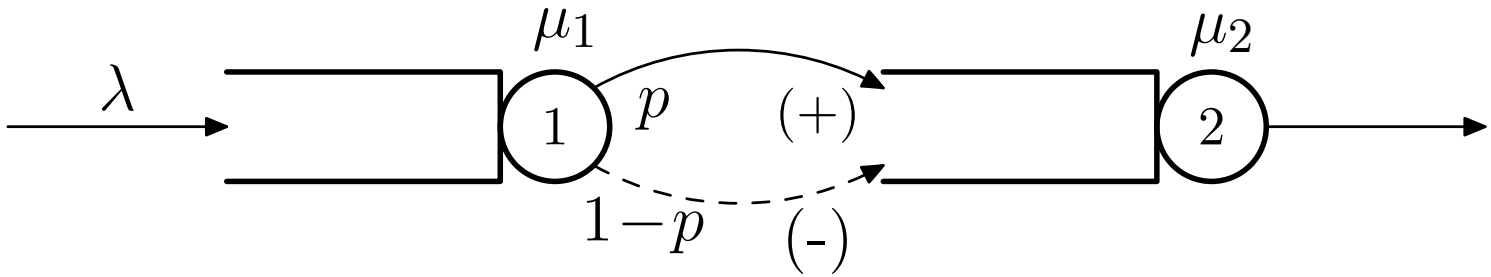
The Markov graph associated with the basic G-queue depicted above.

The dynamics

In words,

- positive customers behave as ordinary ones;
- negative customers are actually signals: when a negative unit arrives at the queue, if there is some positive customer there, it removes it from the system (both "disappear"), and if the queue is empty, nothing happens;
- only positive customers can be observed; for negative ones, only their action can;
- "G-queue" and "Random Neuron" are two names for the same mathematical object;
- there are two different vocabularies, depending on the context: "G-queue" $\equiv$ "Random Neuron"; "backlog of the queue" $\equiv$ "potential of the neuron"; "busy queue" $\equiv$ "active or excited neuron"; etc.

From G-queues to G-networks



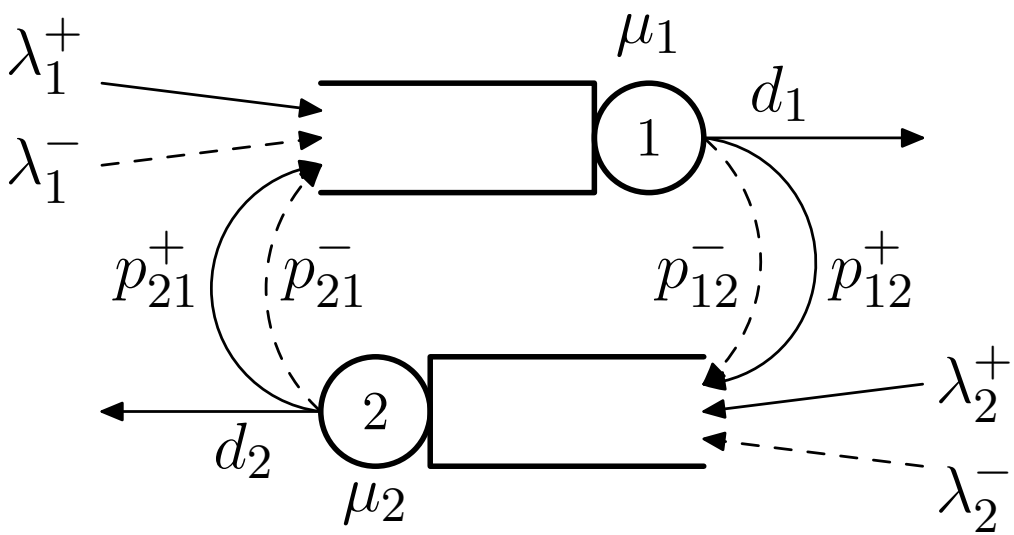
A tandem: first queue is an M/M/1; when leaving the first node, with probability p customers go to a second queue as positive ones, and with probability $1-p$ as negative signals; service rate at queue i is μ_i, $i = 1, 2$.

Here, theory says that stability happens $\iff$ $\lambda < \mu_1, \mu_2$, and in that case, if $X_i = 1($ queue i is busy at $\infty\ )$, $i = 1, 2$, we have, for $j, k = 0, 1$:

$$\Pr(\text{ at } \infty,\ X_1 = j,\ X_2 = k\ ) = \left[(2\rho_1 - 1)j + 1 - \rho_1\right]\left[(2\rho_2 - 1)k + 1 - \rho_2\right]$$

(these models are product form queueing networks).

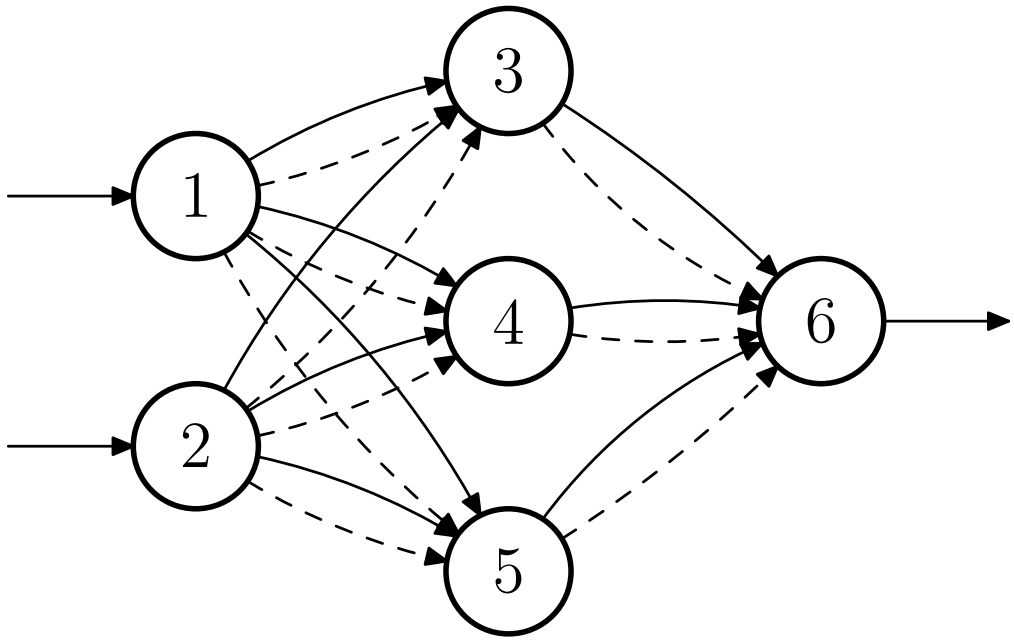
A recurrent G-network



A general recurrent G-network with two nodes. We denote by d_i the probability of leaving the network after a service at queue i.

The asymptotic independence of the indicators of busy neurons is valid in general, so also here (and potentials are also asymptotically independent).

A 3-layer structure

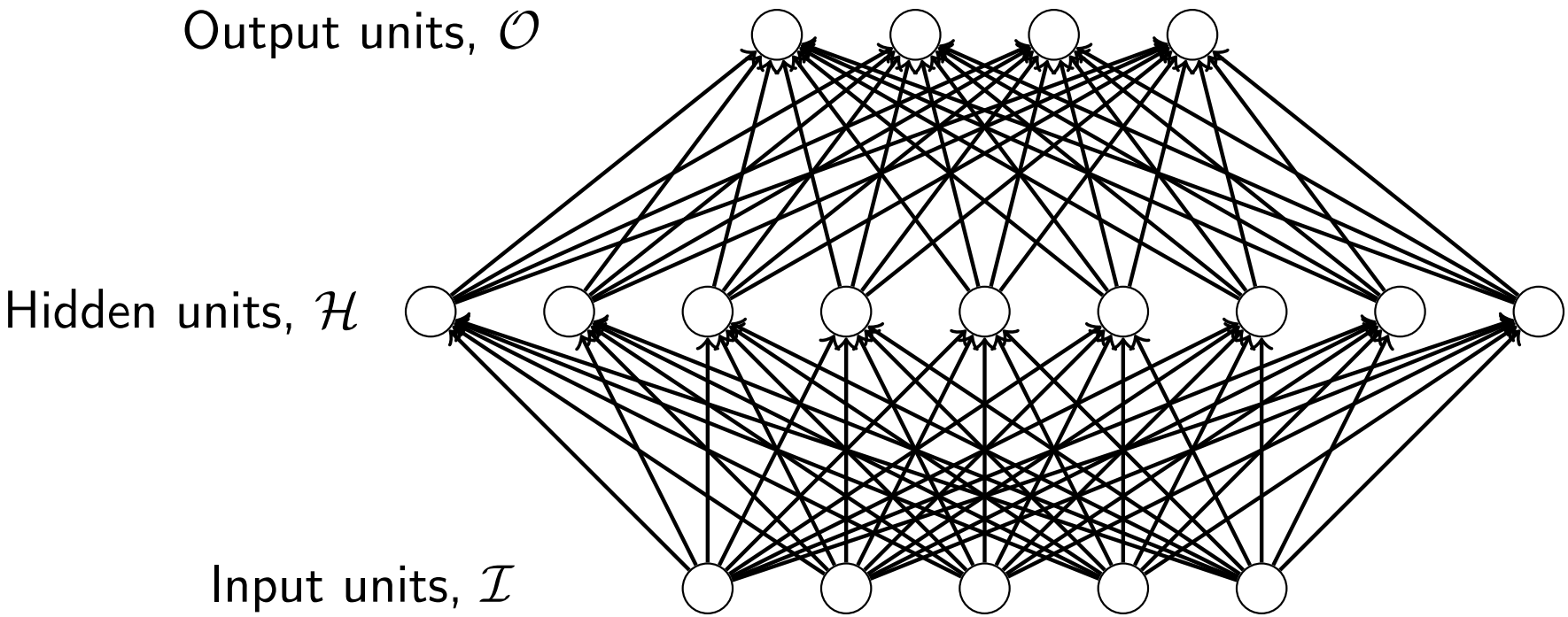


The graph of a 3-layer RNN of the (2,3,1) type (as before, dashed arrows correspond to flows of negative customers/signals).

On the weights of a RNN

- Think queueing: when a customers leaves queue i (necessarily a positive customer), it goes to queue j as a positive customer with some fixed probability $p^+_{i,j}$, or as a negative customer with some fixed probability $p^-_{i,j}$.
- Then, the mean throughput (the frequency) of the flow of positive customers (of spikes) from i to j is $r_i p^+_{i,j} =: w^+_{i,j}$, and of negative ones is $r_i p^-_{i,j} =: w^-_{i,j}$.
- Since for any neuron i in the network, we must have $\sum_j \left(p^+_{i,j} + p^-_{i,j}\right) = 1$ (add a queue/neuron 0 representing the network's environment), we have $\sum_j \left(w^+_{i,j} + w^-_{i,j}\right) = r_i$.

The general 3-layer Random Neural Network



Network inputs: those flows arriving at the input neurons; network outputs: the flows leaving the output neurons.

Random Neural Networks implement rational functions

- Assume the negative ports *of input neurons* aren't used. Assume a single output neuron, so, a scalar network output.
- Call x_i the signal arriving at the positive port of input neuron i. Then, we can explicitly write the network output as a function of the inputs.

$$z_o = \frac{\sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^-_{i,h}} w^+_{h,o}}{r_o + \sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^-_{i,h}} w^-_{h,o}}.$$

- The output is thus a rational function of the input. This allows many treatments. Also, for learning, costs (errors) are rational functions of weights, also leading to simple and powerful treatments.

Outline

A video evaluation PSQA function

- In one of our works for the Perceptual Quality analysis of video sequences (for specific network characteristics, coding, etc.), we built a PSQA function $\nu()$ with 5 variables:
  - BR, the bit rate,
  - FR, the frame rate,
  - RL, a redundancy parameter of MPEG,
  - LR, the loss rate at the packet level,
  - MLBS, the mean size of loss bursts.
- After validation, we obtained that the quality Q of the video sequences as perceived by the users is accurately represented by a function $\nu(BR, FR, RL, LR, MLBS)$, a ratio between two polynomials in these five variables.
- This function is associated with other specific technical parameters that don't change, so, they are not input variables (e.g., the codecs used).

Standard modeling approach

- Assume that we want to do some end-to-end performance evaluation of an app transporting video.
- Then assume that the network (or its bottleneck) is represented by some classic stochastic model. For instance, assume for illustration purposes, the following simple (and unrealistic) assumptions:
  - the video flow arrives as a Poisson process with rate FR fps to the queue,
  - which has a transmission rate of c bps;
  - frames have an average size of B bits; the size is seen as a real number;
  - the storage capacity is HB bits.

Standard modeling approach (cont'd)

- Standard approach: since quality strongly depends on the loss process, focus on the loss probability.
- Looking at the model at the frame level, this is an $M/M/1/H$ queue. Arrival rate: FR. Service rate: c/B. Capacity: H.
- Result: the loss probability is

$$\mathrm{LR} = \frac{1-\rho}{1-\rho^{H+1}}\rho^{H},$$

with $\rho = \dfrac{\mathrm{FR}}{c/B}$, assumed here to be $\neq 1$.

Using PSQA

Proposed approach:

- In the $M/M/1/H$ model, a simple Markovian analysis gives: $\mathrm{MLBS} = 1 + \rho$.
- The quality as perceived by the user, as a function of the data BR, FR, RL, c and H is then

$$\nu\left(\mathrm{BR}, \mathrm{FR}, \mathrm{RL}, \frac{1-\frac{\mathrm{FR}}{c/B}}{1-\left(\frac{\mathrm{FR}}{c/B}\right)^{H+1}}\left(\frac{\mathrm{FR}}{c/B}\right)^{H}, 1+\frac{\mathrm{FR}}{c/B}\right).$$

This is a "formula" giving the *ultimate* target, the quality as perceived by the user, as a function of the data (= BR, FR, RL, c and H).

Variant in VoIP

Similarly (the goal here is to use the same queueing model) consider the bottleneck of a VoIP connexion represented by a $M/M/1/W$ queue.

PSQA variables:

- source bit rate (in Kbps),
- length of the packets (in msec),
- FEC (Forward Error Correction) offset,
- network loss rate,
- network mean loss burst size.

Using PSQA: example in VoIP (cont'd)

For instance,

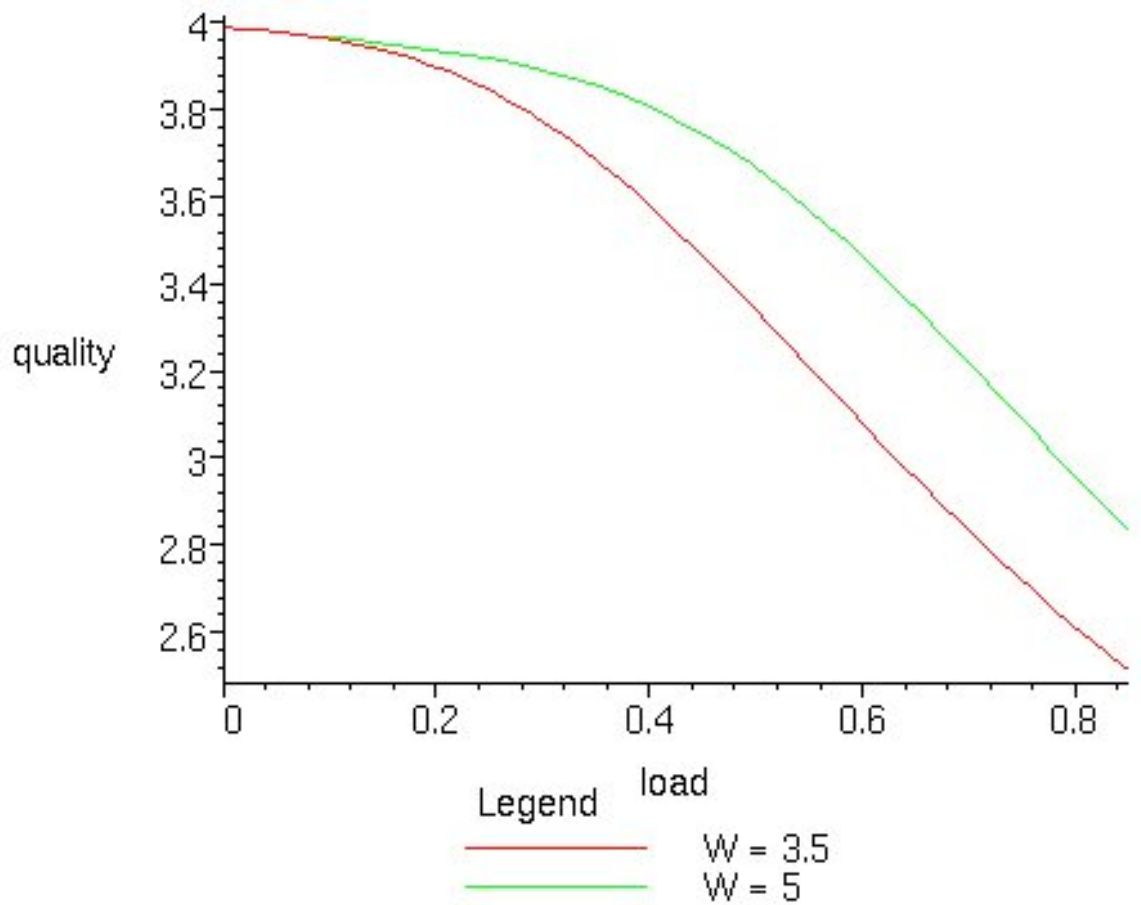
- using the PCM codec,
- FEC with an offset of 1
- and a packetization interval equal to 20 ms,

leaving only free the window size W and the load ρ, and using a pretty simple neural architecture, the perceived quality is well represented by

$$Q = \frac{\alpha + \beta\rho + \gamma\rho^W + (\gamma + \alpha)\rho^{W+1} + \beta\rho^{W+2}}{\alpha' + \beta'\rho + \gamma'\rho^W + (\gamma' + \alpha')\rho^{W+1} + \beta'\rho^{W+2}},$$

where $\alpha = 0.1326$, $\beta = 0.0201$, $\gamma = 0.0674$, $\alpha' = 0.1659$, $\beta' = 0.0399$, $\gamma' = 0.9326$.

Using PSQA: example in VoIP (cont'd)



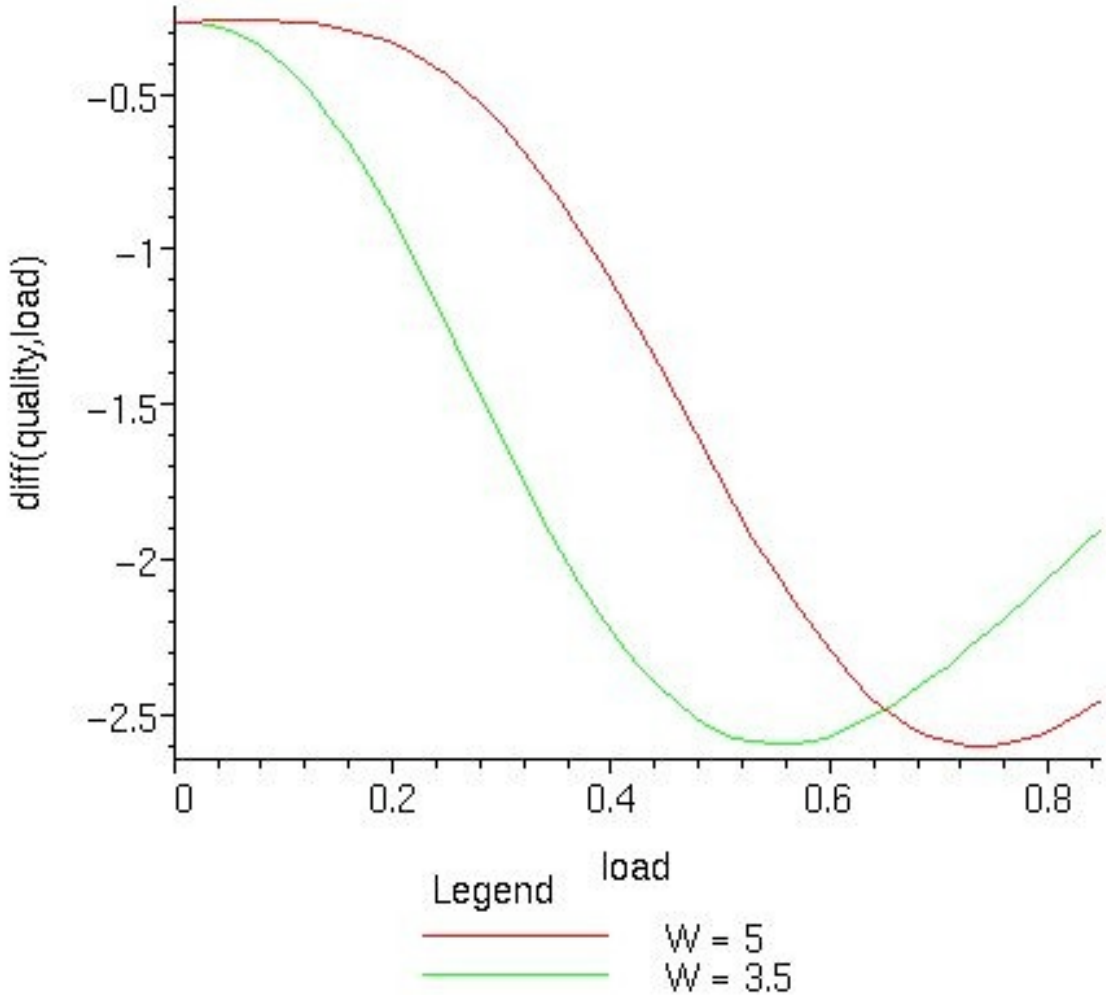
Quality as a function of network load,
for two values of W, with FEC (offset = 1).

Using PSQA: example in VoIP (cont'd)

- For instance, using $W = 5$, we can look for the maximal load we can accept in order to have the perceived quality at least equal to 0.7. We obtain $\rho = 0.4354$.
- If the buffer size is $W = 10$, the maximal load is $\rho = 0.6425$.

Using PSQA: example in VoIP (cont'd)

Having analytical expressions allows for other mathematical treatments. For instance, here is an example of sensitivity analysis (quality with respect to load):



Utility functions

- In many analysis, we must use (postulate) utility functions $U()$ representing the gain the user obtains from the network, as a function of some measured resource, for instance, bandwidth.
- Let c denote bandwidth and $U = U(c)$ the utility function.
- $U()$ is assumed to have some mathematical properties (e.g. concavity, existence of derivatives at any order, etc.).
- Using the previous approach and the fact that $\nu()$ (that is, the RNN model) has nice mathematical properties ($\nu()$ is a rational function), we can work with specific and meaningful utility functions.

Utility functions (cont'd)

- In the previous example, assume we use a very simple RNN with only no "hidden layer".
- In this case, $\nu()$ is the ratio between two polynomials in the selected 5 variables having both degree one.
- If we need to look at the quality as a function of the bandwidth, we can fix BR, FR, RL, and W, and after some algebra, we obtain

$$Q = \frac{(\alpha + \beta c + \gamma c^2 - \delta c^{W+1} - \phi c^{W+2})(\psi - c^{W+1})}{(\alpha' + \beta' c + \gamma' c^2 - \delta' c^{W+1} - \phi' c^{W+2})(\psi' - c^{W+1})}$$

where α, β, ..., ψ' are positive constants.

Utility functions (cont'd)

- Since we use normalized variables (everything is scaled to be in $[0, 1]$, so $c < 1$), we used the approximation

$$Q \approx \frac{\psi}{\psi'} \frac{\alpha + \beta c + \gamma c^2}{\alpha' + \beta' c + \gamma' c^2},$$

which was accurate enough. This can be written

$$Q \approx K \frac{1 + Ac + Bc^2}{1 + A'c + B'c^2},$$

etc.

Outline

Standard modeling approach

- Standard situation: evaluation of some dependability metric associated with a service (or more generally, with some specific aspect of the network).
- Examples of typical metrics: having defined what's an operating system is (user-oriented definition),
  - reliability at $t = R(t) = \Pr(\text{the system works from 0 to } t)$,
  - MTTF = Mean Time To Failure = $\mathrm{E}(\text{system's life-time}) = \int_0^\infty R(s)\,\mathrm{d}s$,
  - Point Availability at $t = PAV(t) = \Pr(\text{the system works at time } t)$,
  - Interval Availability on $[0, t]$ = fraction of $[0, t]$ during which system works,

$$IAV(t) = \frac{1}{t}\int_0^t 1(\text{system works at } s)\,\mathrm{d}s,$$

  - Mean Time To Repair, ...

- Assume again the network (or its bottleneck) is represented by a standard simple queue, using the unrealistic assumptions
  - the video flow arrives as a Poisson process with rate FR fps to the queue,
  - which has a transmission rate of c bps;
  - frames have an average size of B bits;
  - the storage capacity is HB bits.

- Assume that the failure of some network component or components makes that the communication is stopped without losses, until the component is back to operation.
- Assume we can represent this making the server of the queue representing the bottleneck of the communication fail.
- Assume exponential service and repair times with respective rates f and r. There is no job losses because of failures, only supplementary delays.
- Looking at the model at the packet level, this is a $M/M/1/H$ with failures and repairs, a Markov chain that can be easily analyzed.

- A typical dependability metric here is the asymptotic availability of the system,

$$PAV(\infty) = \Pr(\text{server is "up"}) = \frac{1}{1+\psi}, \quad \psi = \frac{f}{r}.$$

- A typical "performability" ("performance + dependability") metric could be a conditional asymptotic availability informally defined as the probability that the server is up *when I need it*, that is, when there is some work for it, that is,

$$\Pr(\text{server up} \mid \text{system busy}), = \frac{1}{1+\psi}\frac{\rho(1-\rho^H)}{1-\rho^{H+1}}.$$

Coupling with PSQA

- We had an expression of the perceived quality as a function of 5 parameters, two of them related to the network, the loss probability LR and the mean loss burst size MLBS.
- The direct analysis of the Markov chain allows to derive the loss probability $\mathrm{LR}(\rho, H, \psi)$.
- The computation of the MLBS is much more involved and the result is complex; it is based on the analysis of sojourn times in Markov models and of Palm measures (say, looking at the Markov model at specific points in time). Denote by $\mathrm{MLBS}(\rho, H, \psi)$ the obtained result.

- As a result, we have (an accurate approximation/representation of) the quality as perceived by the user, as a function of the data BR, FR, RL, c, H and ψ:

$$\nu\left(\mathrm{BR}, \mathrm{FR}, \mathrm{RL}, \mathrm{LR}(\rho, \psi), \mathrm{MLBS}(\rho, \psi)\right).$$

Outline

Some local references

- Maths behind our main application:
  "*Quantifying the Quality of Audio and Video Transmissions over the Internet: The PSQA Approach*", G. Rubino, in Design and Operations of Communication Networks: A Review of Wired and Wireless Modelling and Management Challenges, edited by J. Barria, Imperial College Press, 2005.
- Practical aspects of our uses of RNN in learning:
  "*Evaluating Users' Satisfaction in Packet Networks Using Random Neural Networks*", G. Rubino, P. Tirilly and M. Varela, Springer-Verlag Lecture Notes in Computer Science, no. 4132, 2006.
- An example of using RNNs in combinatorial optimization:
  "*A GRASP algorithm with RNN-based local search for designing a WAN access network*, H. Cancela, F. Robledo and G. Rubino, Electronic Notes in Discrete Mathematics 18 (1), 59–65, December 2004.

- An example of application of PSQA:
  "*Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric*", J.-M. Bonnin, G. Rubino and M. Varela, in The Computer Journal, 49(2):137–155, 2006.
- On the design of a P2P streaming network based on PSQA:
  "*A robust P2P streaming architecture and its application to a high quality live-video service*", H. Cancela, F. Robledo Amoza, P. Rodríguez-Bocca, G. Rubino and A. Sabiguero, in Electronic Notes in Discrete Mathematics 30: 219–224, 2008,
  plus another paper with a demo,
  "*Automatic Quality of Experience Measuring on Video Delivering Networks*", D. De Vera, P. Rodríguez-Bocca and G. Rubino, in SIGMETRICS Performance Evaluation Review, Vol. 36, Issue 2, associated with a demonstration at Sigmetrics'08 awarded with the Best Demonstration Prize.

- An example of improvement on the initial RNN tool:
  "*Levenberg-Marquardt Training Algorithms for Random Neural Networks*", S. Basterrech, S. Mohammed, G. Rubino and M. Soliman, in The Computer Journal, Vol. 54, N. 1, 125–135, 2011.
- An example of extension of the initial RNN tool:
  "*Echo State Queueing Networks: a combination of Reservoir Computing and Random Neural Networks*", S. Basterrech and G. Rubino, in Probability in the Engineering and Informational Sciences, Vol. 31, No. 4, pp. 1–16, 2017.

Outline

Summary

- The idea here is to couple some relevant performance model of the application or service centered on the transport of video, audio, voice content, with a PSQA measuring module.
- The model can go from, say, a simple queue to a complex stochastic structure that can only be analyzed using simulation. It has some input variables $x_1, \ldots, x_m$.
- The PSQA $\nu()$ function has its input variables $y_1, \ldots, y_n$. Some of them coincide with the model's x_is. Assume that $x_i = y_i$ for $i = 1, \ldots, k$, with $k < m, n$.
- Then, the analyst's effort is to be able to obtain the remaining $y_{k+1}, \ldots, y_n$ as a function of $x_1, \ldots, x_m$.
- The result is a new function $Q(x_1, \ldots, x_m)$ where Q is the QoE. This worked several times in our previous works.

Current and future work

- Random Neural Networks are, at least, interesting competitors in the ML field.
- We have extensions of the RNN model to predict time series, for a PSQA 2.0 project.
- We have worked on improving the learning procedures with RNNs, and we are exploring deep architectures now.
- We are preparing a project for avoid using human panels in the process (leading to big data problems).
- We have another project to accelerate the analysis of complex performance models that target QoE, using the extrapolating capabilities of RNNs.